```cpp
#include <iostream>
#include <fstream>
#include <vector>

struct Point {
    int id;
    double x;
    double y;
};

class DataReader {
public:
    DataReader(const std::string& filename) : filename(filename) {}

    bool readData() {
        std::ifstream file(filename);
        if (!file.is_open()) {
            std::cerr << "Error opening file: " << filename << std::endl;
            return false;
        }

        std::string line;
        while (std::getline(file, line)) {
            if (line == "PTS") {
                readPoints(file);
            } else if (line == "CON") {
                readConnections(file);
            }
        }

        file.close();
        return true;
    }

    const std::vector<Point>& getPoints() const {
        return points;
    }

    const std::vector<std::pair<int, int>>& getConnections() const {
        return connections;
    }

private:
    void readPoints(std::ifstream& file) {
        points.clear();
        std::string line;
        while (std::getline(file, line) && line != "CON") {
            Point point;
            if (std::sscanf(line.c_str(), "%d %lf %lf", &point.id, &point.x,
&point.y) == 3) {
                points.push_back(point);
            }
        }
    }

    void readConnections(std::ifstream& file) {
        connections.clear();
        std::string line;
        while (std::getline(file, line)) {
```

```
            int from, to;
            if (std::sscanf(line.c_str(), "%d %d", &from, &to) == 2) {
                connections.push_back(std::make_pair(from, to));
            }
        }
    }

    std::string filename;
    std::vector<Point> points;
    std::vector<std::pair<int, int>> connections;
};

int main() {
    DataReader reader("data.txt");

    if (reader.readData()) {
        const std::vector<Point>& points = reader.getPoints();
        const std::vector<std::pair<int, int>>& connections =
reader.getConnections();

        std::cout << "Points:" << std::endl;
        for (const Point& point : points) {
            std::cout << "ID: " << point.id << ", X: " << point.x << ", Y: " <<
point.y << std::endl;
        }

        std::cout << "Connections:" << std::endl;
        for (const auto& connection : connections) {
            std::cout << "From: " << connection.first << ", To: " <<
connection.second << std::endl;
        }
    }

    return 0;
}

```